CLAIMS

1. A graphics processor for multithreaded execution of program instructions associated with threads to process at least two sample types comprising:

a thread control unit including:

a thread storage resource configured to store thread state data for each of the threads to process the at least two sample types, wherein a number of locations in the thread storage resource allocated for a sample type is determined using a sample portion global state value.

2. The graphics processor of claim 1, wherein the sample portion global state value is a fixed value.

3. The graphics processor of claim 1, wherein the sample portion global state value is programmed.

4. The graphics processor of claim 1, wherein the sample portion global state value is determined dynamically.

5. A graphics processor for multithreaded execution of program instructions associated with threads to process at least two sample types comprising:

a thread control unit including:

a thread storage resource configured to store thread state data, the thread state data including a write flag.

6. A method of multithreaded processing of graphics data comprising:

receiving a sample;

determining a type of the sample; and

assigning the sample to a thread for processing.

7. The method of claim 6, further comprising:

determining that a thread is available for assignment to the sample.

8. The method of claim 6, further comprising:

determining that a thread is available for assignment to the type of the sample.

9. The method of claim 6, further comprising:

passing a priority check based on an allocation priority for the sample type prior to assigning the sample to the thread.

10. The method of claim 6, further comprising:

receiving another sample;

determining that a thread is not available for assignment to the other sample; and

waiting for a thread to become available for assignment to the sample.

11. A method of multithreaded processing of graphics data comprising:

determining that a thread is available for assignment to a sample;

determining that a sample is available to be processed by a thread;

determining a type of the sample that is available; and

assigning the sample that is available to the thread that is available.

12. The method of claim 11, further comprising:

determining that additional samples are available to be processed by threads; and

using a thread allocation priority to identify a sample for assignment to the thread that is available.

13. The method of claim 11, further comprising:

determining that a thread is not available for assignment to another sample; and

waiting for a thread to become available for assignment to the sample.